A World Wide Web Resource Discovery System

Budi Yuwono Savio L. Lam Jerry H. Ying Dik L. Lee
Department of Computer Science
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong

Abstract

As the WWW grows at an increasing rate, efforts to make the technology more manageable are highly in demand. Applying advanced information retrieval techniques is one approach to such efforts. Despite the potential benefit of these techniques in reducing users information overload and improving the effectiveness access to on-line information, little research has been done on applying them to WWW environment. In this paper we present our attempt to apply the vector space retrieval model, relevance feedback mechanisms and a hypertext mapping technique as an integrated resource discovery system to the WWW. This paper discusses some design issues involved, as well as practical issues such as retrieval effectiveness, usability and scalability.
Keywords: resource discovery, information retrieval, world wide web indexing, user interface

1 Introduction

The explosive growth of the World Wide Web (WWW) [4] is making it difficult for a user to locate information that is relevant to his/her interest. There are just too many servers to access and pages to browse through, even keeping track of new information as it becomes available on-line is a very time consuming task. Many methods have been proposed to alleviate this so called internet resource discovery problem. In general, most methods employ keyword searching strategy which allows a user to locate WWW page addresses or URL's (Uniform Resource Locator [1]) by specifying a keyword or keywords. Given the keywords, such a system searches into its index and, upon finding hits, it then shows a list of URL's with their respective titles, possibly ranked by their relevance scores. This search strategy complements browsing or hypertext navigation, the access method of the WWW, by providing users with potentially relevant starting points for browsing.

As with other applications on the Internet that are distributed in nature, the WWW is highly diversified and is ever changing in a non-coordinated way. An ideal search tool should

be able to keep track of such changes, detect any inconsistencies caused by independent local modifications, and organize information into a structured index which enables efficient searches. Such an index can substantially reduce network load since users need to access only the index data in order to locate resources. In this paper we present a WWW resource discovery system which employs a so called Web robot to build and maintain an index database for keyword searching. This approach, as opposed to manual indexing, is suitable for the size and the dynamical nature of WWW environment.

The issue in the design of a keyword-based search tool is how effective the tool can meet the user's information needs. This involves the choice of the search algorithm and the user-system interaction component. In our previous work, we have studied a number of document search and ranking algorithms for WWW environment. From our empirical evaluation, we concluded that TFxIDF algorithm which is based on word distribution outperforms other algorithms which rely on WWW hyperlink information [14].

No matter how good the search and ranking algorithm is, however, it does not guarantee that the documents or WWW pages assigned high relevance scores are actually relevant to the user's information need. Such discrepancies may occur due to ill-defined queries, such as too few/many keywords used resulting in not specific/general enough queries, problems related with synonyms and homonyms, or the user's unfamiliarity with the subject he/she is interested in. One of the approaches to alleviating this problem is by means of a relevance feedback mechanism, i.e., an interaction between the user and the system to iteratively refine a query through a subsequent number of retrievals until a desired set of retrieved documents is obtained. In this paper, we present a relevance feedback mechanism which is supported by the gateway component of our system. The mechanism expands the previous version of a query by adding certain keywords extracted from WWW pages marked as relevant by the user.

There is, however, a problem with the above scheme, namely judging which WWW pages are relevant among those retrieved by the search engine requires the user to access and read through the actual pages, which can be time consuming. Even if the user has the time and patience to do so, it may not be easy to fully make sense of the contents of the individual pages without examining the context at the neighboring pages. The reason is that in hypertext environments such as WWW's, a document or page is often only a small chunk of a larger discourse formed by a network of interlinked pages. Our system's gateway component alleviates this problem by presenting the search result in a hierarchical page map which shows the links among the retrieved pages and links with the neighboring pages.

While different users may have different information needs, it is often desired to share one user's discovery with other users who may find it interesting. To facilitate such information sharing, our index server allows any user to save his/her query statement on the server's side so that other user or the user him/herself can reuse it later. This mechanism can also save other users time to perform query refinement themselves.

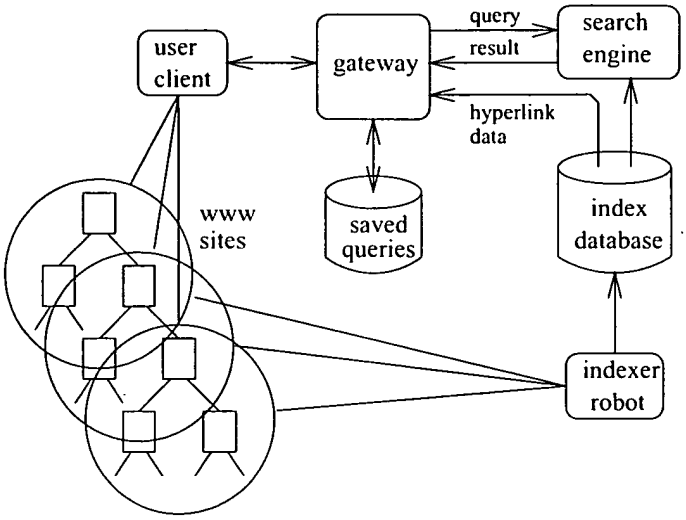

Figure 1: The WWW Resource Discovery System's architecture.

The remaining of the paper is organized as follows. In section 2 we provide a detailed description of the components of our WWW resource discovery system, namely the indexer robot, the search engine and the gateway. Section 3 discusses lessons learned from the trial run of the system, particularly regarding its retrieval effectiveness and usability, and the issue of system scalability. Section 4 closes this paper with a brief comparison with other work and conclusions.

2 System Description

Figure 1 shows the architecture of our WWW resource discovery system [1] The system's main components are the Indexer Robot, the Search Engine, and the Gateway.

2.1 Indexer Robot

The indexer robot is an autonomous WWW browser which communicates with WWW servers using HTTP (Hypertext Transfer Protocol [2]). It visits a given WWW site, traverses hyperlinks in a breadth-first manner, retrieves WWW pages, extracts keywords and hyperlink data from the pages, and inserts the keywords and hyperlink data into an index. A manually prepared list of target sites is used to create the index initially. In order to accommodate the constantly growing number of WWW servers on-line, the system allows any user to register new sites or URL's.

[1]The WWW Resource Discovery System currently covers sites in Hong Kong, and is publicly accessible at ⟨http://dbcl13.cs.ust.hk:8001⟩.

The index consists of a page-ID table, a keyword-ID table, a page-title table, a page modification-date table, a hyperlink table, and two index tables, namely, an inverted index and a forward index. The page-ID table maps each URL to a unique page-ID. The keyword-ID table maps each keyword to a unique keyword-ID. The page-title table maps every page-ID to the page's title. The page modification-date table maps every page-ID to the date when the page was last visited by the indexer robot. The hyperlink table maps each page-ID with two arrays of page-ID's, one array representing a list of pages referencing the page (incoming hyperlinks) and the other array representing a list of pages referenced by the page (outgoing hyperlinks). The inverted index table maps each keyword-ID to an array of (page-ID, word-position) pairs, each of which represents a page containing the keyword and the position of the word (order number) in the page. This word-position information is used in query by phrase or keyword-order specific search and relevance feedback mechanism. The forward-index table maps a page-ID to an array of keyword-ID's representing keywords contained in the page. To obtain a fast access speed, hashing method is used to index each of these tables on the page-ID or keyword-ID attribute.

In extracting the keywords, we exclude high-frequency function words (stop-words), numbers, computer specific identifiers such as file-names, directory paths, e-mail addresses, network host names, and HTML (Hypertext Markup Language [3]) tags. To reduce storage overhead, the indexer robot only indexes words enclosed by HTML tags indicating tokens such as page titles, headings, hyperlink anchor names, words in bold-face, words in italic, and words in the first sentence of every list item. We assume that a WWW author will use these tags only on important sentences or words in his/her WWW pages. Thus, these words make good page identifiers. This is one of the advantages of adopting SGML (Standard General Markup Language), of which HTML is a subset. Words chosen as keywords are then stemmed by removing their suffices.

Resources using protocols other than HTTP (FTP, Gopher, Telnet, etc.) or in formats other than HTML text file (non-inline image, sound, video, binary, and other text files), click-able image maps, and CGI scripts are indexed by the anchor texts referencing them.

Periodic maintenance of the index files is performed bi-weekly by the indexer robot. First, the indexer robot checks the validity of every URL entry in the database by sending a special request to the WWW server containing the page to check whether the page has been modified since the time it was last accessed by the indexer robot. This special request, known as HEAD request, is a feature supported by HTTP. Non-routine index maintenance is also supported. This is performed at night in response to user requests received during the day to index new pages (URL's) or re-index updated pages. Such user requests are facilitated by an HTML form provided by the gateway component (to be discussed later).

The indexer robot has the capability of detecting looping paths, e.g., those caused by Unix symbolic links, and does not visit the same page more than once unless the page has been modified since the time it was last visited by the robot. The latter is made possible by

supplying the last access date and time into the HTTP request [2]. As specified in the HTTP specification [2], the remote WWW server will not send the page content in response to the request if the page has not been modified since the specified time. Furthermore, the indexer robot will not even send an HTTP request if the page was last accessed within the last 24 hours. This is to prevent the robot from sending more than one HTTP requests for the same page during a maintenance batch. To prevent itself from roaming around uncontrollably from one server to the next, the indexer robot accesses one site at a time and only references within the same site domain as that of the referencing page are traversed. Finally, the indexer robot supports the proposed standard for robot exclusion [3] which prevents robots from accessing places where, for various reasons, they are not welcome. The indexer robot is written in the C language. All index files are implemented using the GNU GDBM Database Manager library package [9].

2.2 Search Engine

The search engine employs the so called TFxIDF search and ranking algorithm which assigns relevance scores to documents [4] using the vector space model [12]. The scores indicate the similarities between a given query, represented by a query vector, and the documents, represented by document vectors. In the vector space model, the similarities are measured by the cosine of the angle between the query vector and each of the document vectors. Each component of such a vector corresponds to the weight of a word or term in a query or a document. The weight of a term in a document is a function of the term's term frequency (TF) and the term's inverse document frequency (IDF). Generally speaking, the relevance score of a document is the sum of the weights of the query terms appearing in the document. More formally, the relevance score of document i with respect to query Q is:

$$R_{i,Q} = \sum_{term_j \in Q} (0.5 + 0.5\frac{TF_{i,j}}{TF_{i,max}})IDF_j \tag{1}$$

where $TF_{i,j}$ is the term frequency of term j in document i, $TF_{i,max}$ is the maximum term frequency of a keyword in document i, and IDF_j is the inverse document frequency of term j, defined as:

$$IDF_j = \log(N/DF_j) \tag{2}$$

where N is the number of documents in the collection, and DF_j is the number of documents containing term j.

This term weighting function gives higher weights to terms which occur frequently in

[2] Using the *If-Modified-Since* request header field.

[3] (http://web.nexor.co.uk/users/mak/doc/robots/norobots.html).

[4] The term "documents" here refers to text files in general. It also applies to WWW pages.

a small set of the documents. As shown in equation 1, the term weight is normalized for document length, so that short documents have equal chances to be retrieved as longer documents. In our earlier work [14], it was shown that vector length normalization [13] does not work well for WWW environment.

Given a query, the search engine computes the scores of WWW pages in the index, and returns at most the top H pages, where H is the maximum number of hits. By default H is set to 40, although the user can set it to a different value if so desired. Since the index database contains all of the information needed for ranking, the ranking process does not have to access any WWW pages physically.

The search engine supports Boolean constructs which make use of & (logical AND), | (logical OR), and brackets (scope marker) notations. To accommodate user who are not familiar with Boolean constructs, it allows the keywords to be separated by white spaces, which are treated as logical AND operators. Hyphens may also be used as keyword separators to specify phrases for keyword-order specific searching.

2.3 Gateway

The user interface to the search engine is an HTML form which can be invoked by standard WWW client programs. The form allows the user to type in a query, execute a search, set the maximum number of hits, access documentation pages about our server, access/run sample queries or saved queries, and invoke other HTML forms for registering a URL or writing comments to us. Figure 2 shows a screen dump of the gateway's home page. The version of the gateway shown in figure 2 has an algorithm selection option which allows the user to use a different algorithm other than the default TFxIDF. The modular design of the search engine makes it easy for us to plug in and out new algorithms for experimentation. The client-server interface mechanism is implemented using the standard Common Gateway Interface (CGI) and is run by NCSA HTTPD version 1.3 WWW Server program.

A query begins as the user types in the keywords and clicks on a submit button to send the query to the search engine. Upon receiving the result from the search engine, the gateway displays a list of URL's and their respective titles ordered in descending relevance scores. The user can physically access these URL's by clicking on the titles. In addition to the above ordered list, the result page also shows other information and buttons for various functionalities, which are described in the following paragraphs.

2.3.1 Hierarchical Page Map

In order to help the user identify the content of a WWW page without having to access the physical page, the retrieved pages, represented by their titles, are displayed in a hierarchical

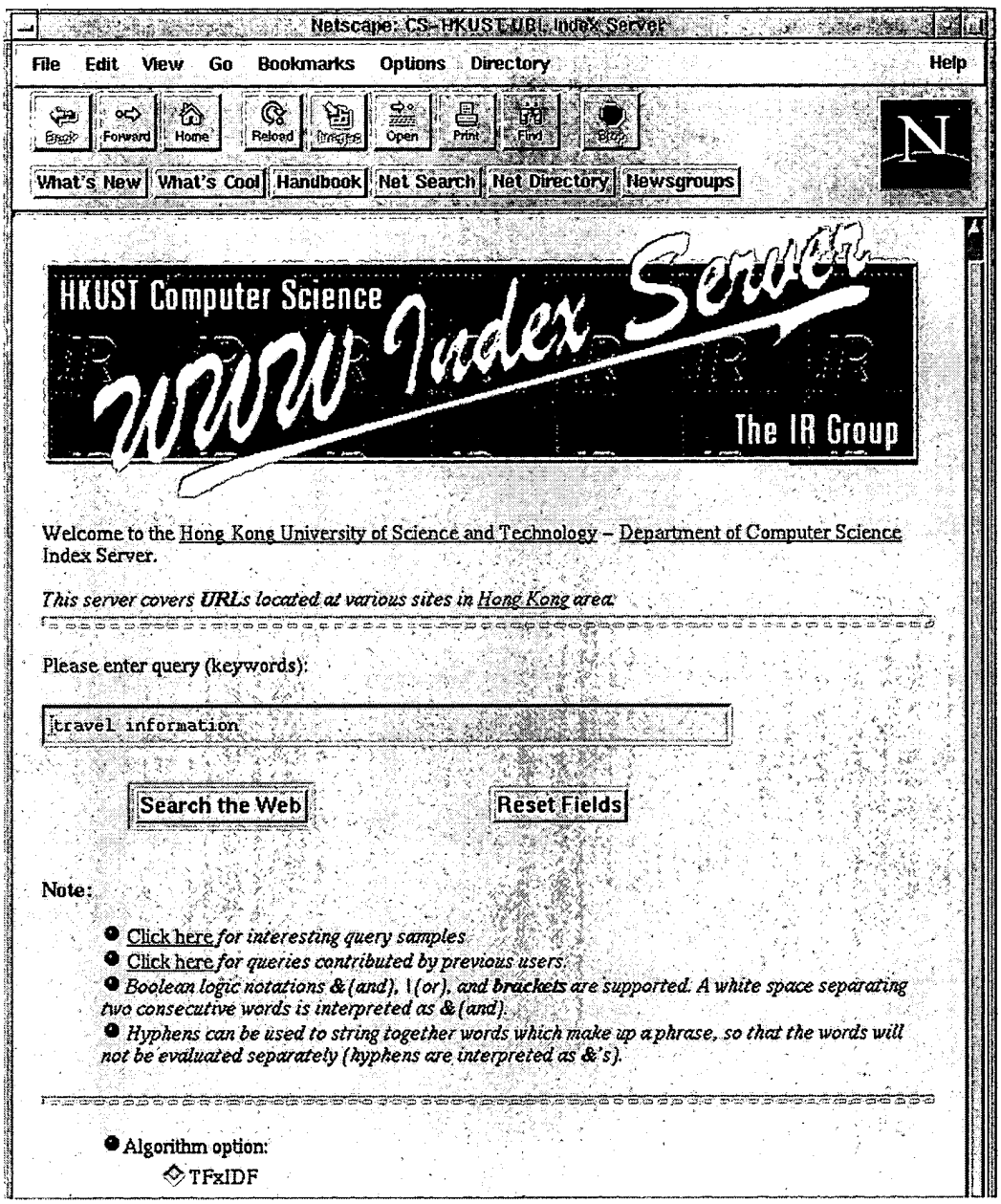

Figure 2: The homepage of the WWW Resource Discovery System's gateway.

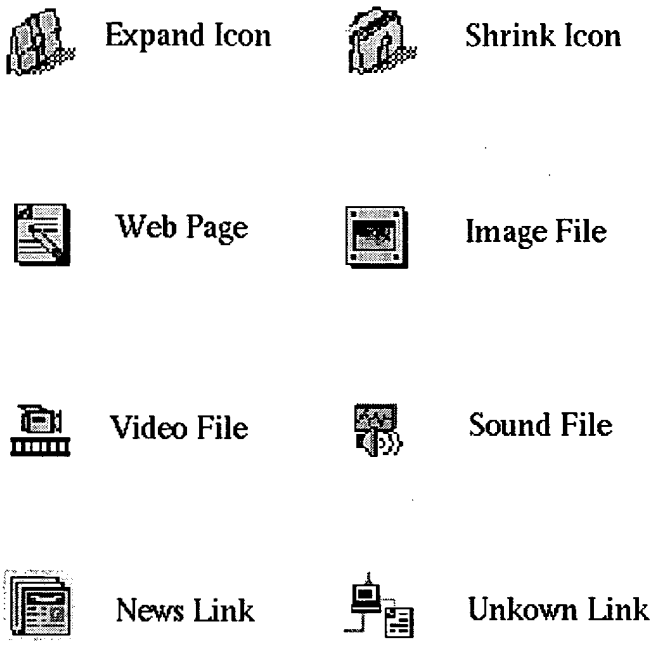

Figure 3: The icons and their meanings.

organization. This organization, shown by the use of line indentations, depicts the parent-child or referrer-referee hyperlink relationships among the pages. Accompanying each of the page titles an icon is shown indicating the page's data format (image file, sound file, video file, etc.) or one of two special icons representing an expandable item (closed suitcase) and a shrinkable item (open suitcase) respectively. Figure 3 shows the page icons and their meanings. When the user clicks on a closed suitcase icon, a new result page is

then displayed with the WWW pages referenced by the page whose icon was clicked on, shown as a list of titles with line indentations under the title of the referencing page. At this point the referencing page is shown with an open suitcase icon. Clicking on this open suitcase icon will revert the result page to its original state, i.e., shrinks the result by removing child pages under the referencing page whose icon was clicked on. This expand/shrink operation is performed by the gateway using the hyperlink data available from the index database. Figure 4 and 5 show an example of such operations.

The user can physically access any of the pages displayed on the result page by clicking on its title. Finally, the gateway keeps track of the user's session state by assigning a unique page-ID to every result page it generates. Every request issued by the user from clicking on a button of a page is tagged with the page's ID. The gateway maintains the information associated with every page-ID for a period of time. This information is purged when its period is up.

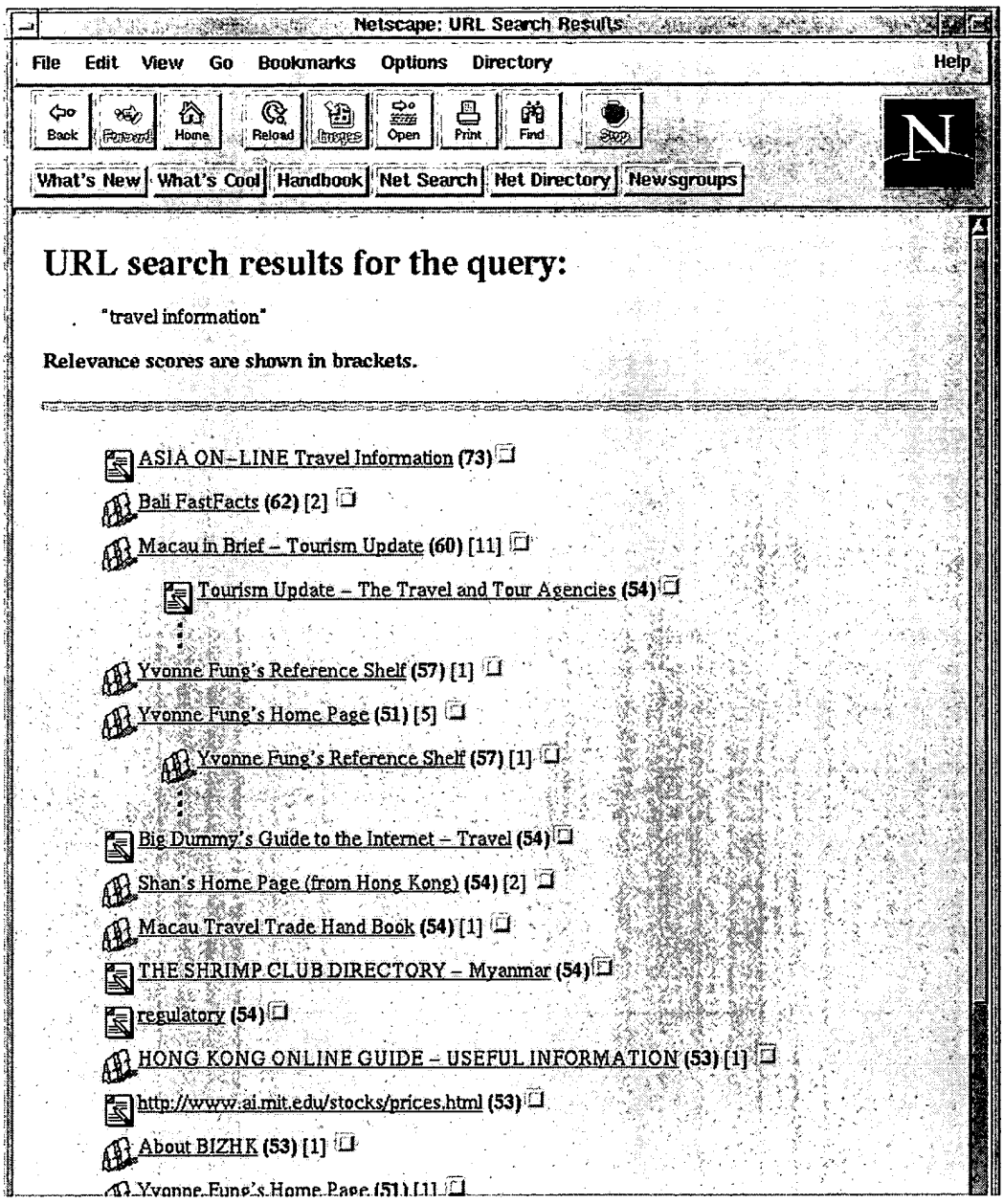

Figure 4: The result page displaying a list of WWW pages ordered in descending relevance scores. Line indentation is used to depict hierarchical dependence relationships among the pages.

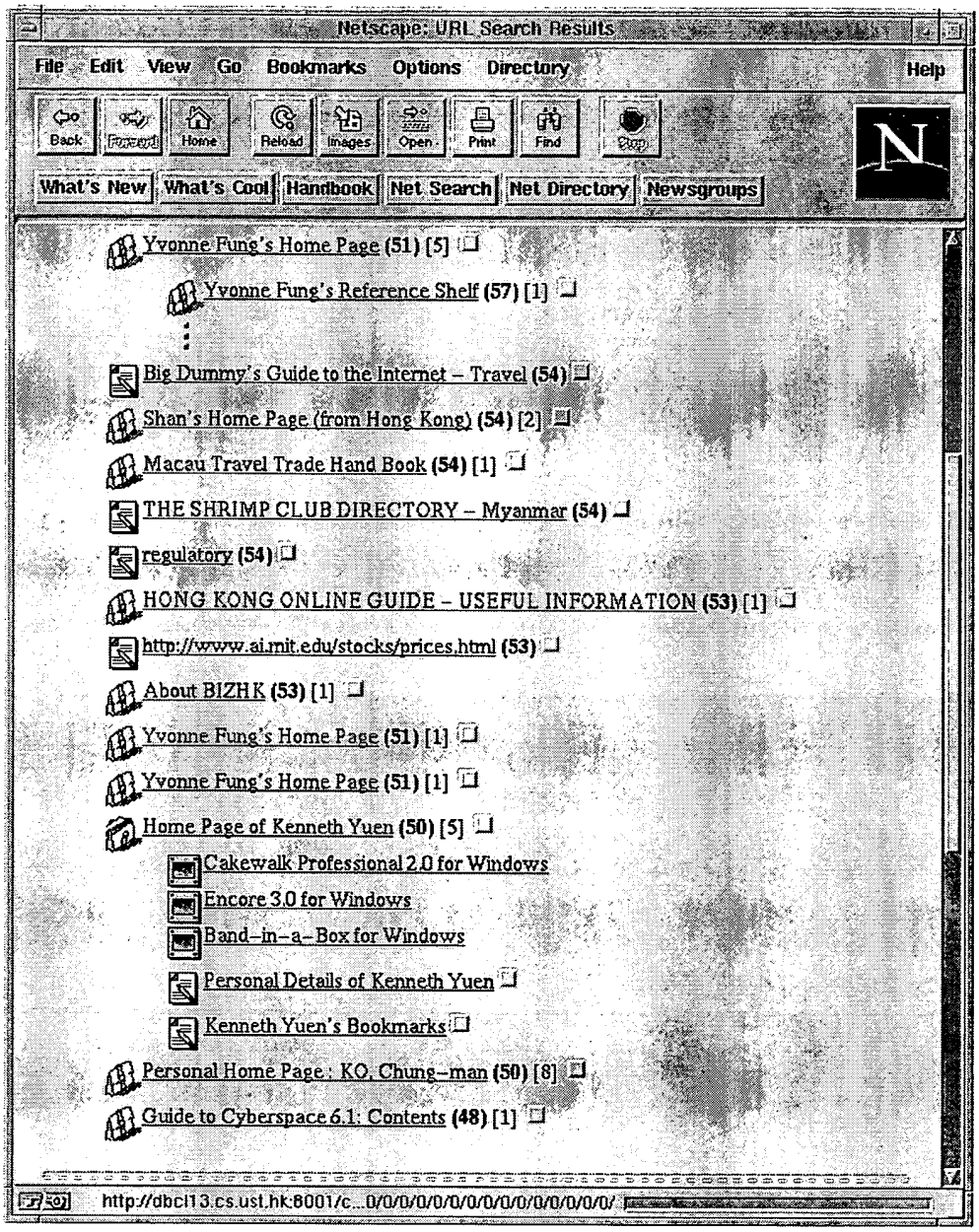

Figure 5: The result page after a page expansion operation (from the result page shown in figure 4).

2.3.2 Relevance Feedback

The remaining part of the result page is an HTML form used for the relevance feedback mechanism (see figure 6).

First, the user is asked to mark WWW pages judged as relevant to his/her query by clicking on a check box button next to each of the pages' titles. Next, the user can select the query expansion algorithm to use, if so desired, by clicking one of the radio buttons under the algorithm option. The algorithms supported are as follows:

- **High**
  This algorithm selects a number of most frequently occuring keywords from the feedback pages, and appends them to the original query.
- **Low**
  Similar to High, but this algorithm selects the least frequently occuring keywords from the feedback pages.
- **Hits**
  A hit is when a word in the original query appears in a feedback page. This algorithm selects a number of keywords surrounding the hit word in the feedback pages, and appends them to the query.

The user is allowed to specify the number of words to select from each of the feedback pages, otherwise, the default value of 5 is used. These three algorithms are designed to refine a query by capturing the context in which the original query words appear in the feedback pages [8].

As in the home page, search options for selecting the search/ranking algorithm and specifying the maximum number of hits are also provided. Finally, the relevance feedback parameters are submitted to the gateway for processing when the user clicks on the **Feedback** button, and another result page is then shown.

The gateway component consists of two separate modules, one for handling page expansion/shrinking operations and the other for handling the relevance feedback mechanism. These modules are written in the C language and run as CGI scripts.

2.3.3 Query Sharing

At the bottom of the result page, a button is provided for the user to save the query, so that others who find it interesting can readily reuse it. When clicked upon, the button invokes an HTML form which allows the user to supply the title of the query, if so desired,

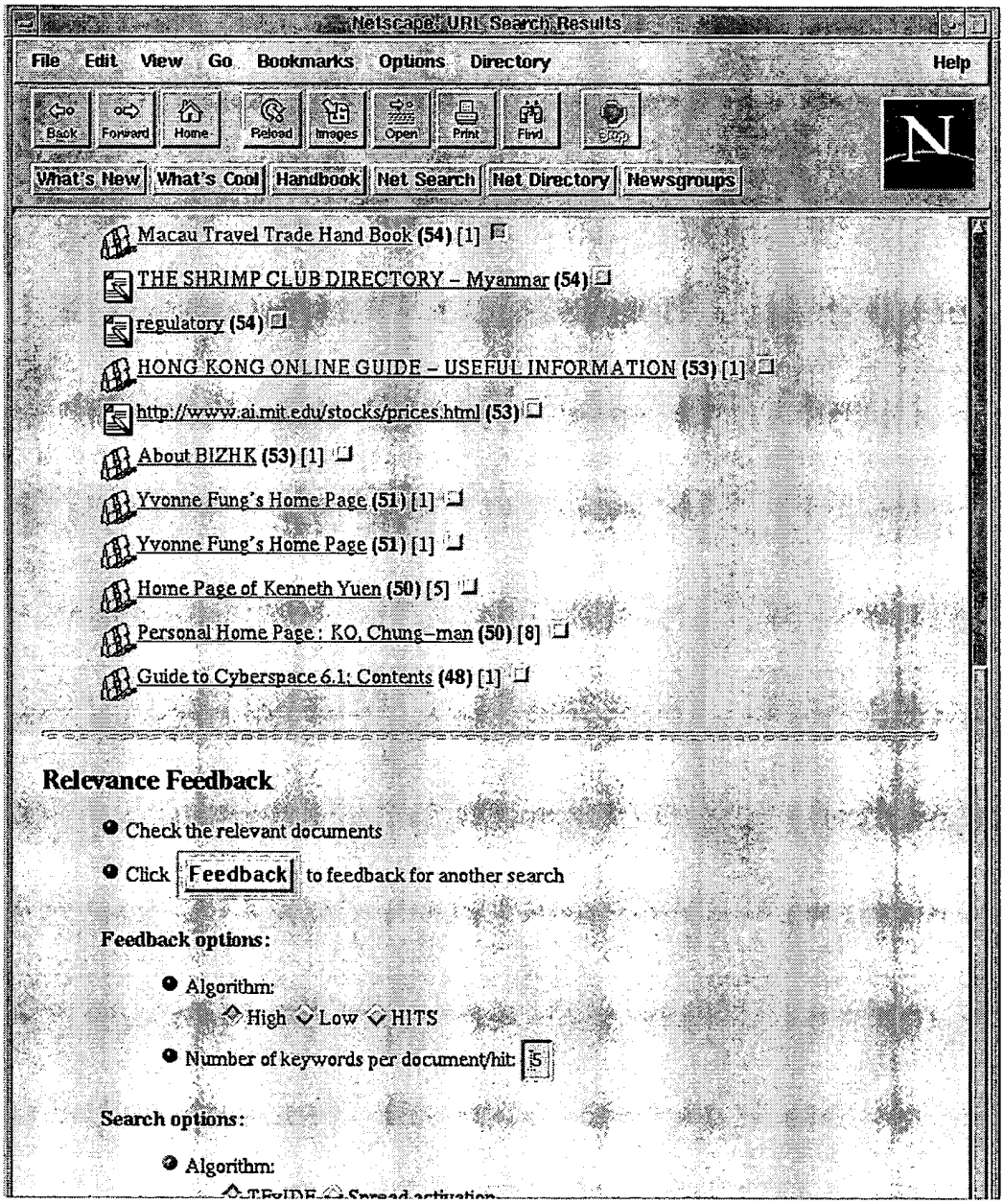

Figure 6: The relevance feedback form.

and submit the query to the gateway. Saved queries are stored in a list of click-able query statements. By clicking on one of these statements, a user can resubmit the query to the search engine.

3 Lessons Learned

3.1 Search Effectiveness

Although fully operational, most components of our system are still in prototype stage where developments and evaluations are still underway. Evaluating an information retrieval system for WWW environment is a difficult task. The difficulty stems from the unavailability of standard test data and also the highly subjective nature of the notion of relevance of WWW pages retrieved with respect to the user's information need. In our previous work, for lack of standard evaluation methods, we empirically proved the effectiveness of TFxIDF search and ranking algorithm using a traditional recall-precision measurement with an actual WWW collection obtained from an academic site in Hong Kong. The reader is referred to our report [14] for the full detail of the study.

3.2 Usability

Another aspect on which the system's effectiveness should be judged is its usability. We have not conducted a formal usability study on our system. However, from our access log file, we learned that the average query length is 1.3 words, or in other words, most queries are single-word ones. While there is nothing wrong with single-word queries, as users often just want to browse around without any specific information need in mind, such a query often results in more pages than the maximum number of hits allows, particularly if the word is a common one. In such a case, the user may not aware of the existence of pages which may be of his/her interest but were cut off by the maximum number of hits.

We also learned from the access log that most users do not attempted to refine their queries, e.g., by adding or dropping keywords, to narrow down or broaden up the scope of their searches when we expected them to do so, at least out of intuition. However, we can not tell what happened without any verbal protocol data. It may be that the user lost interest in the search, or the user already found what he/she was looking for, or our system was too clumsy to interact with, or the user was not familiar enough with the keyword searching process to be able to fine tune the query. The latter pessimistic view may also be the reason why, as the log file showed, not many users try to use our relevance feedback mechanism. This observation is not something new to those who work with systems which employ advanced information retrieval techniques. Even constructing a boolean query with logical notations is not something that an average user can be expected to do.

3.3 Hypertext Mapping

One of the critical issues in hypertext navigation is the user disorientation which may be experienced as the user traverses a densely entangled network of hyperlinks. One of the commonly proposed solutions is the use of graphical maps which display the information as networks with documents as the nodes and hyperlinks as the edges. We consider our hierarchical page maps as one step toward realizing such an approach.

However, the main design objective of our hierarchical page maps is to provide efficient means for the user to identify a WWW page by showing its context, represented by links to the neighboring pages, and the data format of the page, without having to physically access the page and its neighboring pages. This approach can help preserve the network's bandwidth and save the user time to load the pages, especially on slow networks or from busy servers.

Our hierarchical page maps use page titles, for lack of better representation, to represent the contents of WWW pages. The problem with this scheme is that many page titles do not reflect the contents of the pages they represent. Also, it is not uncommon to find a group of pages, usually within the same directory path, to have the same title. Still some other pages do not have titles for various reasons. Obviously, a better page representation scheme is needed to make this approach effective. One possible alternative is to use keywords. Another related design issue is how to make the representation concise yet meaningful enough to be displayed, possibly in graphical format, as network nodes in a hypertext map.

3.4 Scalability

Another important issue that needs to be addressed for a resource discovery system to be useful, is its scalability. With the current rate of growth of the WWW, indexing a large number of WWW sites will be no longer practical. As of this writing, the overall size of our index database which covers almost all sites in Hong Kong is still manageable at about 10 MBytes. Ideally, a WWW resource discovery system should cover all sites around the world, which is what other WWW index servers such as WebCrawler, WWWW, Lycos, etc., do. However, maintaining the validity and updating such a large database is not practical and will put too much burden on the physical network. The problem becomes worse when such a maintenance is performed by many index servers with overlapping coverages.

For the above reason, we strongly advocate a system of distributed index servers with minimal overlapping coverages, for example by regional division. We are currently working on a mechanism which facilitates interchange of meta information among autonomous distributed index servers such that each of these servers can redirect a query to another server which can potentially gives a better result. This approach is similar to that of other server indexing methods such as GLOSS [6] and directory of servers in WAIS [7]. Upon receiving

a query, an index server checks whether it should process the query or redirect it to other index servers which can potentially give better results. A simple communication protocol is used by the servers to exchange server keywords with each other.

4 Other Work and Conclusions

4.1 Other Index Servers

There are many robot-based WWW index and search services on the Internet today[5]. However, among the well known robots, only a few employ full-text indexing, e.g., WebCrawler [6] [11], the Repository Based Software Engineering Project Spider [7] (RBSE) [5], and Lycos [8]. Other services index only page titles and first level headings (e.g., JumpStation [9]), or titles, headings and anchor hypertexts (e.g., World Wide Web Worm or WWWW [10]). Our indexer robot takes other HTML tokens such as words in bold-face or italics, the first sentence of every list item, in addition to titles, all-level headings and anchor hypertexts. Our scheme is a balance between full-text and title-only schemes by taking advantage of HTML meta-information as much as possible. On a WWW page containing mostly lists such as an index page, our scheme extracts nearly as much words as a full-text scheme.

Not many index servers use sophisticated information retrieval models beyond a simple Boolean model or pattern matching based on Unix *egrep* (e.g., WWWW), with exception of the RBSE, the WebCrawler, and Lycos. The RBSE uses WAIS search engine which ranks WWW pages based on the occurrence frequencies or term frequency (TF) of the query words [10. The WebCrawler and Lycos, as with our search engine, rank the pages based on term frequency and inverse document frequency (TFxIDF).

4.2 Conclusions

As the WWW grows at an increasing rate, efforts to make the technology more manageable are highly in demand. Applying advanced information retrieval techniques is one approach to such efforts. Despite the potential benefit of these techniques in reducing users information overload and improving the effectiveness access to on-line information, little research has been done on applying them to WWW environment. The work presented in this paper is but an early stage of such research. We believe that there is still room for improvements

[5] A list of WWW robots can be found at ⟨http://web.nexor.co.uk/mak/doc/robots/active.html⟩.
[6] ⟨http://webcrawler.cs.washington.edu/WebCrawler/Home.html⟩.
[7] ⟨http://rbse.jsc.nasa.gov/eichmann/urlsearch.html⟩.
[8] ⟨http://lycos.cs.cmu.edu/⟩.
[9] ⟨http://www.stir.ac.uk/jsbin/js⟩.
[10] ⟨http://www.cs.colorado.edu/home/mcbryan/WWWW.html⟩.

and many strategies yet to be explored. For instance, search strategies which take advantage of WWW-specific meta information such as the semantic of hyperlinks may offer a better retrieval performance. Finally, the other important issue that may have been hindering the application of advanced information retrieval techniques to the WWW environment, or any other on-line information environment for that matter, is usability. This calls for better user interface designs which can make these techniques more intuitive for average users to use.

5 References

1. Berners-Lee, T., "Uniform Resource Locators," *Internet Working Draft*, 1 January 1994.

2. Berners-Lee, T., "Hypertext Transfer Protocol," *Internet Working Draft*, 5 November 1993.

3. Berners-Lee, T., and Connolly, D., "Hypertext Markup Language," *Internet Working Draft*, 13 July 1993.

4. Berners-Lee, T., Cailliau, R., Groff, J., and Pollermann, B., "World Wide Web: The Information Universe," *Electronic Networking: Research, Applications and Policy*, 1(2), 1992.

5. Eichmann, D., "The RBSE Spider - Balanching Effective Search against Web Load," In *Proceedings of the First International Conference on the World Wide Web*, Geneva, Switzerland, May 1994.

6. Gravano, L., Tomasic, A., and Garcia-Molina, H., "The Efficacy of GLOSS for the Text Database Discovery Problem," Technical Report STAN-CS-TR-93-2, Stanford University, October 1993.

7. Kahle, B., and Medlar, A., "An Information System for Corporate Users: Wide Area Information Servers," Technical Report TMC-199, Thinking Machines, Inc., April 1991.

8. Lee, D. L., and Chuang, A. H., "Performance of Document Ranking and Relevance Feedback," submitted for publication, 1994.

9. Nelson, P., "GDBM - The GNU Database Manager," online manual pages, Version 1.7.3, 1990.

10. Pfeifer, U., Fuhr, N., and Huynh, T., "Searching Structured Documents with the Enhanced Retrieval Functionality of freeWais-sf and SFgate," *Computer Networks and ISDN Systems*, 27(7), pp. 1027-1036, 1995.

11. Pinkerton, B., "Finding What People Want: Experiences with the WebCrawler," In *Proceedings of the First International Conference on the World Wide Web*, Geneva, Switzerland, May 1994.

12. Salton, G., *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley, Reading MA, 1989.

13. Salton, G., and Buckley, C., "Term-Weighting Approaches in Automatic Text Retrieval," *Information Processing &Management*, 24(5), pp. 513-523, 1988.

14. Yuwono, B., and Lee, D. L., "Search and Ranking Algorithms for Locating Resources on the World Wide Web," submitted for publication, 1995.